



TECHNICAL REPORT

YR-2007-001

CONSTRUCTING A MAXIMUM UTILITY SLATE OF ON-LINE ADVERTISEMENTS

S. Sathiya Keerthi and John A. Tomlin

Yahoo! Research

2821 Mission College Blvd.

Santa Clara, CA 95054

{selvarak, tomlin}@yahoo-inc.com

April 4, 2007

Santa Clara, California • Berkeley, California • Burbank, California • New York, New York
Barcelona, Spain • Santiago, Chile

Yahoo! Research Report No. YR-2007-001

Yahoo! Research Report No. YR-2007-001

CONSTRUCTING A MAXIMUM UTILITY SLATE OF ON-LINE ADVERTISEMENTS

S. Sathiya Keerthi and John A. Tomlin
Yahoo! Research
2821 Mission College Blvd.
Santa Clara, CA 95054
{selvarak, tomlin}@yahoo-inc.com

April 4, 2007

ABSTRACT: We present an algorithm for constructing an optimal slate of sponsored search advertisements which respects the ordering that is the outcome of a generalized second price auction, but which must also accommodate complicating factors such as overall budget constraints. The algorithm is easily fast enough to use on the fly for typical problem sizes, or as a subroutine in an overall optimization.

Keywords: advertising, budgets, column generation, sponsored search

1. Introduction

In this paper we consider the problem of constructing a maximum utility “slate” of ads for display in response to either a search term, or content page requesting ads from a server. We shall treat both of these essentially the same—that is that some ordered set of ads (the “slate”) is to be returned in response to a “query”.

Let us denote the n bidders on this query by $j = 1, \dots, n$, and suppose that some form of Generalized Second Price auction (GSP)[7] has been carried out, which induces a ranking of the bidders. For simplicity, let the numerically ordered indices also indicate the bid ranking, initially assumed to be determined solely by the bids—sometimes called the Overture ranking¹. That is, if we denote the bid of bidder j by A_j , then

$$A_1 \geq A_2 \geq \dots \geq A_n.$$

Let m be the maximum number of positions, and let T_{jp} be the click-through rate (CTR) of bidder j when his ad is at position p . Finally, let ρ_j be a “utility factor” associated with the appearance of bidder j ’s ad in response to the query.

We define a slate as an ordered subset $S = \{j_1, \dots, j_k\}$, where $k \leq m$, of the ordered set of ads $\{1, \dots, n\}$. Since we are initially assuming a second-price auction by bid value, bidder j_p in position p pays the bid $A_{j_{p+1}}$ of the bidder occupying position $p+1$. In addition there is a minimum bid ϵ , which is paid by the last bidder in the slate if and only if there are no lower bidders on the query. Under these assumptions we wish to solve

$$\max_S U = \sum_{p=1}^k \rho_{j_p} T_{j_p p} A_{j_{p+1}} \quad (1)$$

subject to the requirement that j_1, \dots, j_k is an increasing set of indices, and $k \leq m$.

Assuming the CTRs are independent, the utility U of the slate when the ρ_j are all unity is easily seen to be the expected revenue from the slate. However there may be several reasons why we wish to consider other values of the ρ_j . Some of the more important of these are:

1. The advertiser placing the ad may be at, or near, its budget, thus reducing the desirability of showing it. This is our primary motivation, and this is a companion paper to [2], which discusses this in detail. For convenience, we include a brief description of this model in an Appendix.
2. “Ad Fatigue” induced by too-frequent showing of an ad may reduce its effectiveness. We might therefore wish to penalize some ads which have been shown above some threshold.
3. We may be given (or wish to have) the CTRs as a product two components - a component solely due to the ad, independent of position, and a position-only dependent component. In this case we may reduce T_{jp} to a position only component T_p and an ad-dependent component which becomes a contributor to ρ_j . However, this is not a necessary feature and we will usually continue refer to the CTRs as T_{jp} .

¹This assumption will later be generalized.

Yahoo! Research Report No. YR-2007-001

The treatment we give here is independent of the source of the ρ_j .

There is some superficial similarity with the well-known knapsack problem[9]—we are selecting a subset S of up to m items (the ads), subject to constraints. Also related is the “knapsack auction” considered in [3]. Even more strongly related is the cardinality constrained knapsack problem[6], since our slates have a fixed maximum size. However, the ordering requirement makes the problem more specialized. Nevertheless, like the knapsack problem, our problem is amenable to a dynamic programming approach.

2. Backward Recursion

We begin by giving a $O(n^2m)$ algorithm for solving (1) using a dynamic programming (DP) algorithm with backward recursion[5]. For this approach, it is convenient to include m dummy bidders, call them $n + 1, \dots, n + m$, with $\rho_j = 0$ and $A_j = \epsilon$ for $j = n + 1, \dots, n + m$. The reason for doing this is that we only need to consider slates of size equal to m ; smaller slates can be padded with dummy bidders at the end to produce the same effect and make a slate of size exactly m .

Take one j and one s such that $1 \leq s \leq m$. Let us define a *subslate* starting from j at position s as a set of increasing indices, $\tilde{S} = \{j_s, j_{s+1}, \dots, j_m\}$ such that $j_s = j$. Let $\tilde{S}(s, j)$ denote the set of all such subslates. Consider the problem of computing the best “marginal-revenue-to-go”:

$$F(j, s) = \max_{\tilde{S} \in \tilde{S}(s, j)} \sum_{p=s}^m \rho_{j_p} T_{j_p p} A_{j_{p+1}} \quad (2)$$

Suppose we fix j_{s+1} and proceed optimally from there. If $F(j_{s+1}, s + 1)$ is known for all $j_{s+1} > j$ then we can compute (2) in standard dynamic programming fashion as:

$$F(j, s) = \max_{j_{s+1} > j} \rho_j T_{j s} A_{j_{s+1}} + F(j_{s+1}, s + 1) \quad (3)$$

We can start the DP algorithm by setting $F(j, s) = 0 \forall j = n + 1, \dots, n + m, s = 1, \dots, m$. Now recurse backwards and compute $F(j, s) \forall s = 1, \dots, m$ and $j = n, \dots, 1$. Finally, choose $\max_j F(j, 1)$ to get the solution of (1) as well as the optimal slate.

3. An Optimal Path Approach

Frequently, problems that are amenable to dynamic programming can be cast in the form of a shortest or longest path problem [4], and this is no exception. Let us define a network with nodes $N_{j,p}$ for $p = 1, \dots, m$ and $j = p, \dots, n + 1$. We also define terminal nodes $N_{0,0}$ and $N_{n+1,m+1}$. The

3 Slots and 5 Ads (n=5, m=3)

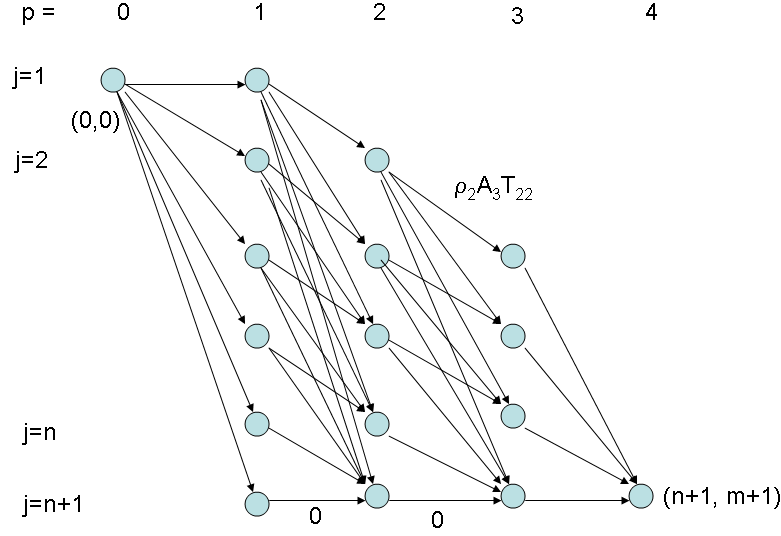


Figure 1: Network with $n > m$

directed edges and their associated costs are defined as:

$$\begin{aligned}
 (N_{0,0}, N_{j,1}) & : c_{0,j,0} = 0 \\
 & \quad (j = 1, \dots, n+1) \\
 (N_{i,p}, N_{j,p+1}) & : c_{i,j,p} = \rho_i A_j T_{ip} \\
 & \quad (j > i \geq p = 1, \dots, m-1) \\
 (N_{j,m}, N_{n+1,m+1}) & : c_{j,n+1,m} = \rho_j A_{j+1} T_{jp} \\
 & \quad (j = p, \dots, n-1) \\
 (N_{n,m}, N_{n+1,m+1}) & : c_{n,n+1,m} = \rho_n \epsilon T_{nm} \\
 (N_{n+1,p}, N_{n+1,p+1}) & : c_{n+1,n+1,p} = 0 \\
 & \quad (p = 1, \dots, m)
 \end{aligned}$$

where $c_{i,j,p}$ is the cost for the edge directed from $N_{i,p}$ to $N_{j,p+1}$ and ϵ is the minimum bid as before. Note that not all these edges are defined (or need to be defined) if $n < m$. (See figures 1 and 2).

Since the network is directed and acyclic, and the utilities/distances associated with the non-trivial edges correspond to the utility of placing ad i in slot p , followed by ad j in slot $p+1$, it is easy to see that the longest path from $N_{0,0}$ to $N_{n+1,m+1}$ maximizes (1) - or equivalently the shortest path using the negatives of the costs defined above.

3 Slots and 2 Ads (n=2, m=3)

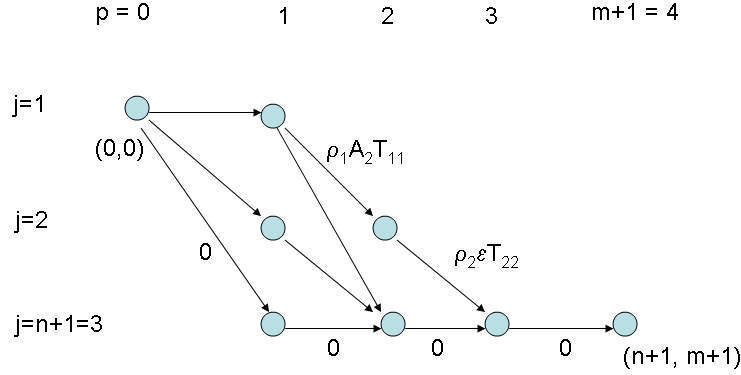


Figure 2: Network with $n < m$

Very efficient algorithms are known for the shortest path problem, but in our case the problem is small, so a simpler implementation suffices.

Since the forward recursion/ optimal path approach is more intuitive and visually appealing, we shall use it for the remainder of this paper in the discussion of extensions and variations.

4. Extension to Revenue Ranking

The present scheme extends to what is sometimes known as revenue ranking, where the ads are ranked not just by bid, but by “expected revenue” which is modeled as the product of the advertiser’s bid A_j and Q_j , which is the “quality score”, or “clickability”, for bidder j ’s ad for this query. This quantity is thought to better represent the value of a bid than the raw bid.

The ads are now ranked according to this product, so that:

$$A_1 Q_1 \geq A_2 Q_2 \dots \geq A_n Q_n.$$

To preserve the condition that the expected payment for a click is at least that of the next ranked bidder, we require that the expected cost per click (CPC) of bidder j_p must be at least $A_{j_{p+1}} \frac{Q_{j_{p+1}}}{Q_{j_p}}$.

Yahoo! Research Report No. YR-2007-001

Using this observation, the path technique extends to the expected revenue ranking scheme. In this case the objective function is now:

$$\text{Maximize } \tilde{U} = \sum_{p=1}^m \rho_{j_p} A_{j_{p+1}} \frac{Q_{j_{p+1}}}{Q_{j_p}} T_{j_p p} \quad (4)$$

where The network model above remains the same except that the edge costs are now modified to be:

$$\begin{aligned} (N_{0,0}, N_{j,1}) &: c_{0,j,0} = 0 \\ &\quad (j = 1, \dots, n+1) \\ (N_{i,p}, N_{j,p+1}) &: c_{i,j,p} = \rho_i A_j \frac{Q_j}{Q_i} T_{ip} \\ &\quad (j > i \geq p = 1, \dots, m-1) \\ (N_{j,m}, N_{n+1,m+1}) &: c_{j,n+1,m} = \rho_j A_{j+1} \frac{Q_{j+1}}{Q_j} T_{jp} \\ &\quad (j = p, \dots, n-1) \\ (N_{n,m}, N_{n+1,m+1}) &: c_{n,n+1,m} = \rho_n \epsilon T_{nm} \\ (N_{n+1,p}, N_{n+1,p+1}) &: c_{n+1,n+1,p} = 0 \\ &\quad (p = 1, \dots, m) \end{aligned}$$

5. Further Extensions

The path technique extends to other practically useful variants. Two of these include introducing restrictions on the subset of ads which may be omitted from the slate, and use of a hybrid objective function which is made up of a weighted sum of the first and second price utilities.

5.1. Restricted Omissions

The algorithm(s) we have been considering assume that any appropriate ordered subset of the ads may be chosen which fits within the slate size. In practice this may not always be true. While it is obviously legitimate to exclude ads from the limited space when the bids (or expected revenues) are too low, it is not so obvious that this may be done for other reasons. For example, one of the motivations we cited for using non-unit weights ρ_j was the need to accommodate bidders with limited budgets. One means of doing this is to allow budgeted bidders to be held out of the notional auction—that is excluded from the slate. However, it is not obvious that this option should extend to unbudgeted bidders. This must be a business decision. We therefore require a means of specifying which ads (bidders) can be excluded for reasons other than low rank. We accomplish this by specifying a *mask* or bit vector, which has a 1 if the ad can be excluded and a zero otherwise. We then modify the algorithm as follows:

Since each arc in the network gives the utility of including a particular ad i in position p followed by ad j , we consider only arcs such that:

1. For each position p we allow i to assume the values from p up to the first ad in rank order which has a zero mask bit. Any subsequent ads are ignored for this p . This ensures that

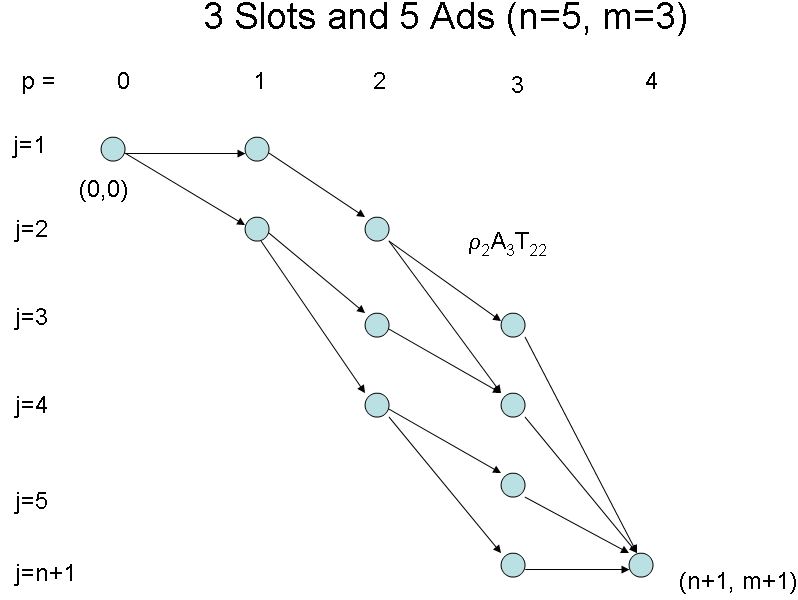


Figure 3: Reduced network with $mask = (1, 0, 1, 0, 1)$

the unmasked ad with the highest rank is not excluded, but that lower ranked ads which are masked are not considered for the position p .

2. For each i chosen as above, the second index j shall only run from $i + 1$ through the next unmasked ad. This ensures that if an ad i can be followed by an unmasked ad it will be the next in rank order.

This scheme may be thought of as actually removing arcs from the networks such as those shown in the figures, or more simply implemented by modifying the longest path algorithm with the rules we have itemized. In Figure 3 we show the reduced network obtained from Figure 1 when we specify that $mask = (1, 0, 1, 0, 1)$. In practice however, we use the second technique of modifying the algorithm.

5.2. Hybrid Objectives

Thus far we have assumed some form of generalized second price auctions is implicit in the utility of the slate. However, even if the price per click is computed with such an assumption, we may wish to include other factors in our utility calculation. For example we may wish to consider

Yahoo! Research Report No. YR-2007-001

the first prices, on the assumption that in a truthful setting these are the actual values placed by the bidders on a click for their ad and we wish to take this into account. Alternatively we may be interested in the raw number of expected clicks. Both of these situations can be accommodated by considering a composite weighted objective function which takes into account both first and second prices in specifying the arc costs in the longest path algorithm. Let us define the hybrid objective as:

$$\text{Maximize } \hat{U} = \sum_{p=1}^m \mu_{j_p} A_{j_p} T_{j_p p} + \sum_{p=1}^m \rho_{j_p} A_{j_{p+1}} \frac{Q_{j_{p+1}}}{Q_{j_p}} T_{j_p p} \quad (5)$$

which may be re-written:

$$\text{Maximize } \hat{U} = \sum_{p=1}^m (\mu_{j_p} A_{j_p} + \rho_{j_p} A_{j_{p+1}} \frac{Q_{j_{p+1}}}{Q_{j_p}}) T_{j_p p} \quad (6)$$

Then if we rewrite the relevant arc costs as:

$$\begin{aligned} c_{i,j,p} &= (\mu_i A_i + \rho_i A_j \frac{Q_j}{Q_i}) T_{ip} \\ &\quad (j > i \geq p = 1, \dots, m-1) \\ c_{j,n+1,m} &= (\mu_j A_j + \rho_j A_{j+1} \frac{Q_{j+1}}{Q_j}) T_{jp} \\ &\quad (j = p, \dots, n-1) \end{aligned}$$

we may solve the optimal path problem as before.

This framework is very flexible and can lead to many extensions. For example if we wish to have a weighted combination of expected revenue and expected clicks we may set the μ_j to $1/A_j$.

Our colleague Zoë Abrams has also informed us[1] that the dynamic programming approach to column generation extends to the Vickery-Clark-Groves (VCG) auction mechanism.

6. Computational Results

All of the algorithms we have described are efficient in terms of number of operations, especially since the numbers involved are relatively small in the on-line advertising framework. Typically m is less than or equal to about 12, and the number of bidders n which need to be considered for inclusion in a slate is less than 100, and may even be less than m .

We have implemented the forward algorithm in a straightforward way in C++, to be called as a subroutine in the column generation algorithm of [2]. When run on a 32-bit Linux box with a 2.8 GHz Xeon processor, it takes an average of 25 microseconds for a sample of 5000 queries where there are 12 ad positions, and between 1 and 77 candidate ads. This includes setting up the data structures as well as the actual path computation, and is obtained without any attempt to optimize the code other than using the `-O2` option of the `gcc` compiler. We can therefore afford to execute the algorithm repetitively, either in real time in an on-line setting, or repeated many times as a subroutine.

7. Conclusion

Sponsored search auctions have recently received considerable attention, but the subsequent problem of how to implement, or adapt, the outcomes in the presence of complicating factors such as budget constraints appears to have been less well studied. We have shown that this can be accomplished in cases of practical interest by a simple but very fast dynamic programming algorithm.

References

- [1] Z. Abrams. Column generation for truthful pricing mechanisms. *Unpublished Yahoo! Research Memorandum*, 2006.
- [2] Z. Abrams, O. Mendelevitch, and J. A. Tomlin. Optimal delivery of sponsored search advertisements subject to budget constraints. *To appear in Proc. ACM EC'07, San Diego, CA. June, 2007.*
- [3] G. Aggarwal and J. D. Hartline. Knapsack auctions. In *Proceedings of the 17th annual ACM-SIAM symposium on discreet algorithms*, pages 1083–1092, 2006.
- [4] E. M. L. Beale. *Introduction to Optimization*. John Wiley and Sons, Chichester et al., 1988.
- [5] R. E. Bellman and S. Dreyfus. *Applied Dynamic Programming*. Princeton University Press, Princeton, NJ, 1962.
- [6] I. de Farias and G. Nemhauser. A polyhedral study of the cardinality constrained knapsack problem. *Mathematical Programming (Ser. A)*, 96:439–467, 2003.
- [7] B. Edelman, M. Ostrovsky, and M. Schwarz. Internet advertising and the generalized second price auction: Selling billions of dollars worth of keywords. *Second Workshop on Sponsored Search Auctions, Ann Arbor, MI. June, 2006.*
- [8] M. E. Lubbecke and J. Desrosiers. Selected topics in column generation. *Operations Research*, 53(6):1006–1027, 2005.
- [9] S. Martello and P. Toth. *Knapsack Problems – Algorithms and Computer Implementations*. John Wiley and Sons, Chichester et al., 1990.

Appendix

The algorithm(s) described in this paper were developed as a column generating subroutine for the linear programming model (LP) described in [2] for optimizing sponsored search ad delivery subject to budget constraints. The concept of a “slate” is put forward in that paper corresponding to columns of the LP. We may formally state the LP as follows:

Yahoo! Research Report No. YR-2007-001

Indices

- $i = 1, \dots, N$ The queries
- $j = 1, \dots, M$ The bidders
- $k = 1, \dots, K_i$ The slates (for query i)

Data

- d_j The total budget of bidder j
- v_i Expected number of occurrences of keyword i
- a_{ijk} Expected cost to bidder j if slate k is shown for keyword i
- r_{ik} Objective function coefficient for slate k for keyword i

Variables

- x_{ik} Number of times to show slate k for keyword i

Constraints

(Budget)

$$\sum_i \sum_k a_{ijk} x_{ik} \leq d_j \quad \forall j \quad (7)$$

(Inventory)

$$\sum_k x_{ik} \leq v_i \quad \forall i \quad (8)$$

Objective

$$\text{Maximize} \quad \sum_i \sum_k r_{ik} x_{ik}$$

Each column of the LP, that is the a_{ijk} values, corresponds to the expected cost per click (CPC) to budgeted advertisers if their ad is clicked on when slate k is shown for query i . If there are more than a handful of budgeted bidders for a query, the number of possible slates is enormous. We therefore require a method for generating those columns which may be included in the LP optimum solution (the well-known idea of ‘‘column generation’’[8]). When the objective function coefficients are the expected revenue from a slate (i.e. $r_{ik} = \sum_j a_{ijk}$), and the dual values corresponding to the budget constraints are π_j , the subproblem we wish to solve is of the form (4) with $\rho_j = 1 - \pi_j$. If the objective coefficients are the bid values (assumed to reflect a bidders true value for a click), the subproblem is of the form (5) with $\mu_j = 1$ and $\rho_j = -\pi_j$. In either case, the slate generated can potentially improve the LP solution if the value is greater than the dual value (say γ_i) for the i^{th} query volume constraint. See [2] for much greater detail.